**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

| | |
|---|---|
| **In re Patent Application:** Christopher L. Brealey et al. | **Group Art Unit:** 2194 |
| **Application No.:** 10/663,396 | |
| **Filed:** September 23, 2003 | **Examiner:** WU, Qing Yuan |
| **Title:** "Encapsulating and Executing Computer Algorithms" | **Confirmation No.:** 3959 |

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

**Sir:**

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on **March 16, 2009**.

(**X**) The fee for filing this Appeal Brief is **$540.00** (37 CFR 41.20).

( ) No Additional Fee Required.

**(complete (a) or (b) as applicable)**

The proceedings herein are for a patent application and the provision of 37 CFR 1.136 (a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: CFR 1.17(a)-(d)) for the total number of months checked below:

- ( ) one month $130.00
- ( ) two months $490.00
- ( ) three months $1110.00
- ( ) four months $1730.00

( ) The extension fee has already been filed in this application

(**X**) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant had inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **09-0461**/CA920030053US1 the sum of **$540.00**. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account **09-0461**/CA920030053US1 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account **09-0461**/CA920030053US1 under CFR 1.16 through 1.21 inclusive, and any other section in the Title 37 of the Code of Federal Regulations that may regulate fees.

Respectfully submitted,

By: /Steven L. Nichols/
**Steven L. Nichols** (Reg. No.: 40,326)
Attorney/Agent for Applicant(s)
Telephone No.: (801) 572-8066
Date: May 18, 2009

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the Patent Application of

Christopher Lawrence Brealey et al.

Application No. 10/669,396

Filed: September 23, 2003

For: Encapsulating and Executing Computer Algorithms

Group Art Unit: 2194

Examiner: WU, Qing Yuan

Confirmation No.: 3959

**APPEAL BRIEF**

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This is an Appeal Brief under Rule 41.37 appealing the decision of the Primary Examiner dated January 21, 2009 (the "final Office Action" or "Action"). Each of the topics required by Rule 41.37 is presented herewith and is labeled appropriately.

## I. Real Party in Interest

The real party in interest is International Business Machines Corporation ("IBM") having a principal place of business at New Orchard Road, Armonk, NY 10504.

## II. Related Appeals and Interferences

There are no appeals or interferences related to the present application of which the Appellant is aware.

## III. Status of Claims

Claims 5, 16-25 have been previously cancelled without prejudice or disclaimer.

Claims 1-4, 6-15 and 26 are pending in the application and stand finally rejected.

Accordingly, Appellant appeals from the final rejection of claims 1-4, 6-15 and 26, which claims are presented in the Appendix.

## IV. Status of Amendments

A single after-final amendment has been filed just prior to the filing of this Brief to correct a minor informality. Appellant cannot indicate at this time whether that amendment has or will be entered as no such decision has been made. However, the amendment proposed in the single after-final amendment corrects a typographical error and has no bearing on the substantive issues presented by this appeal.

## V. Summary of Claimed Subject Matter

The present application discloses a method of executing a computer algorithm (26), including: executing (S42) a first module (30) encapsulating the computer algorithm (26) except at least one communication operation of the algorithm (26) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*), executing (242) a second module (32) encapsulating the at least one communication operation of the algorithm (26) such that the at least one communication operation is available to the first module (30) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*), and instantiating (S44) at least one data object for encapsulating data (34, 36) communicated between the first module (30) and a communicating partner (28) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*). Each one of the at least one data object is an instance of a data class, and the data (34, 36) communicated between the first module (30) and the communicating partner (28) is accessible by the first module (30) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31, 0036-38*).

The claimed subject matter allows for environment-dependent communication operations to be separately encapsulated in one or more communication modules thereby providing the advantage that "the algorithm module can be used, without modification, in different environments." (*Appellant's specification, ¶¶ 0020-21*). Thus, when the algorithm is executed in a new environment, only a different communication module code suitable for use in the new environment need be provided. (*Appellant's specification*, ¶¶ 0032-33).

Turning to Appellant's specific claims,

Claim 1 recites:

A method of executing a computer algorithm (26), comprising:

executing (S42) a first module (30) encapsulating said computer algorithm (26) except at least one communication operation of said algorithm (26) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*);

executing (242) a second module (32) encapsulating said at least one communication operation of said algorithm (26), such that said at least one communication operation is available to said first module (30) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*); and

instantiating (S44) at least one data object for encapsulating data (34, 36) communicated between said first module (30) and a communicating partner (28) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*), each one of said at least one data object being an instance of a data class, said data (34, 36) communicated between said first module (30) and said communicating partner (28) being accessible by said first module (30) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31, 0036-38*)

.

Claim 15 recites:

A computer readable medium storing thereon computer executable instruction code (*Appellant's specification, ¶¶ 0022-27)*, said code when executed by a processor of a computer causes said processor to:

execute (S42) a first module (30) encapsulating a computer algorithm (26) except at least one communication operation of said algorithm (26) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*); and

execute (242) a second module (32) encapsulating said at least one communication operation of said algorithm (26), such that said at least one communication operation is available to said first module (30) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*),

wherein said second module (32) encapsulates at least one environment-dependent communication operation of said algorithm (26) and is configured to communicate with a communicating partner (28) (*Appellant's specification, ¶¶ 0008, 0010, 0028-31*).

## VI. Grounds of Rejection to be Reviewed on Appeal

The final Office Action raised the following grounds of rejection.

(1) Claims 1-3, 15 and 26 were rejected under 35 U.S.C. § 102(b) as being anticipated by Friedman et al., *Problem Solving, Abstraction, and Design Using C++* ("Friedman").

(2) Claims 4 and 6-14 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Friedman taken alone.

Accordingly, Appellant hereby requests review of each of these grounds of rejection in the present appeal.

## VII. Argument

(1) Claims 1-3, 15 and 26 are patentable over Friedman:

Claim 1:

Claim 1 recites:

> A method of executing a computer algorithm, comprising:
> executing *a first module encapsulating said computer algorithm except at least one communication operation of said algorithm*;
> executing *a second module encapsulating said at least one communication operation of said algorithm*, such that said at least one communication operation is available to said first module; and
> instantiating at least one data object for encapsulating data communicated between said first module and a communicating partner, each one of said at least one data object being an instance of a data class, said data communicated between said first module and said communicating partner being accessible by said first module.

(Emphasis added).

Friedman is directed to an elementary discussion of the principles of object oriented programming using the C++ programming language. As such, Friedman teaches general theories relating to the nature of algorithms and data objects and the practical application of those theories in solving basic programming problems. However, nowhere does Friedman teach or suggest the idea of separate modules encapsulating different portions of an algorithm.

According to the recent Office Action, Friedman's discussion of an exemplary Circle class and member functions of the Circle class read on the "first module encapsulating said computer algorithm except at least one communication operation of said algorithm" recited by claim 1 and that a "driver function to test class Circle" taught by Friedman reads on the "second module encapsulating said at least one communication operation of said algorithm." (Action, pp. 2-3) (citing to Friedman, pp. 514-15 and Fig. 11.8). Appellant respectfully disagrees.

In the first place, it is elementary knowledge in object-oriented programming that a data class is a form of abstraction in which the attributes and functions of a type of data object are defined. Thus, a data class is not an algorithm, nor is a data class analogous to an algorithm. Rather, a data class defines the organization (i.e. abstraction) of data that can be passed to and manipulated by an object-oriented algorithm. Friedman makes the distinction very clear by defining an algorithm as "a list of steps," and pointing out that the illustrative Circle class is "used to represent circle objects." (Friedman, pp. 21 and 510). The terms used in Friedman must be construed and interpreted in light of Friedman's own definitions and their accepted usage in the art, neither of which accords with the Examiner's interpretation of these terms.

As was briefly mentioned above, a data class can define functions that may be performed on an object that instantiates the data class. It appears that the functions defined in Friedman's Circle class may be one source of confusion for the Examiner in misconstruing the Circle class as an algorithm. Appellant notes the well-established object-oriented programming principle that a data class merely sets forth *various* functions that *may* be performed on a data object type, while in contrast an algorithm sets forth *which* functions *will* be performed on the object, the ordering and conditions under which those functions will be performed on the object, and the data parameters that will be used in conjunction with the functions. By way of analogy, the functions defined in a data class set forth the tools that may be used on a data object of that class while an algorithm sets forth a plan of how those tools will be used on specific data objects to accomplish a particular task. The tools (functions defined in a class) should not to be mistaken for the plan (algorithm) of how those tools are to be used.

In response, the final Office Action acknowledges that an algorithm is a list of steps to solve a problem and asserts that "if the problem to be solved is to output the circle attributes as indicated by the driver function, the algorithm would nonetheless includes [sic] the steps of implementing the class circle and calling functions of the class circle by the driver function," and that "since the driver function and the class circle are separately implemented, it clearly satisfied the limitation of having separate modules encapsulating different portions of an algorithm." (Action, p. 7). Appellant respectfully disagrees, noting that the steps referenced by the Examiner—"implementing the class circle" and "calling functions of the class circle" are performed by the driver function and not by the Circle class itself. (Friedman, p. 513). Hence, Appellant maintains the position that the Circle class merely represents a set of tools available to an algorithm implemented by the driver function, and is not a separate encapsulation of that algorithm.

Nevertheless, even if the Circle class and the driver function taught by Friedman *were* separate encapsulations of a single algorithm, Friedman still fails to anticipate the subject matter of claim 1. According to claim 1, the first module "encapsulat[es] said computer algorithm except at least one communication operation of said algorithm." In other words, each step in the algorithm implemented by the first and second modules must be present in the first module, with the exception of one or more communication operations of the algorithm. (Claim 1).

The Circle class in Friedman plainly does not teach or suggest this subject matter. The final Office Action itself supports this notion by acknowledging that it is the driver function, not the Circle class, that "call[s] functions of the class circle." (Action, p. 7). Moreover, Appellant notes that the driver function, not the Circle class, generates and sets parameters for a circle to be

implemented from the Circle class (e.g., position, radius, and color). (Friedman, p. 513). This functionality exhibited by the driver function is not an implementation of a communication protocol between the algorithm and an external process. Rather, the steps of calling functions, generating parameters, and setting parameters are substantial operations in the production of an object of class Circle. Because these steps are not implemented by the Circle class, the Circle simply cannot "encapsulat[e] said computer algorithm except at least one communication operation of said algorithm." (Claim 1).

Likewise, the driver function taught by Friedman does not anticipate the first module recited in claim 1. As already noted above, the driver function calls functions of the Circle class, generates parameters for a Circle object, and passes those parameters to the Circle class to instantiate a Circle object having those parameters. Because the driver function relies substantially on the tools available in the Circle class to generate the Circle object, the driver function does not "encapsulat[e] said computer algorithm except at least one communication operation of said algorithm." (Claim 1).

In addition to its failure to anticipate the first module, both the Circle class and the driver function of Friedman fail to anticipate the second module recited in claim 1, which "encapsulates said at least one communication operation such that said at least one communication operation is available to said first module." (Claim 1). When read in context, claim 1 requires that the second module encapsulate *only* communication operations of an algorithm, and not additional algorithm functionality unrelated to communication protocol with an outside process. As noted above, the Circle class includes several coded functions which are invoked to create or manipulate a Circle object, and are therefore not related to communication operations or

protocol. (Friedman, p. 511). Thus, the Circle class of Friedman *cannot* teach or suggest the second module recited in claim 1.

The driver function taught by Friedman also fails to teach or suggest the second module recited in claim 1 because it fails to encapsulat[e] only the communication operation of said algorithm. As noted above, the driver function of Friedman specifies parameters for a Circle object and invokes the functionality of the Circle class. (Friedman, p. 513). This functionality is more than merely communicatory in nature, since a Circle object cannot be formed without the parameters specified by the driver function. (*Id.*).

In conclusion, as demonstrated above, neither the Circle class nor the driver function taught by Friedman teaches or suggests either the first module or the second module recited in claim 1.

"A claim is anticipated [under 35 U.S.C. § 102] only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 2 U.S.P.Q.2d 1051, 1053 (Fed. Cir. 1987) (emphasis added). See M.P.E.P. § 2131. For at least these reasons, the rejection based on Friedman of claim 1 and its dependent claims should not be sustained.

Claim 2:

Claim 2 recites "wherein said at least one communication operation comprises at least one environment-dependent communication operation of said algorithm." In response, the Office Action cites again to Friedman's "driver function to test class circle" and the Circle class itself, stating that "if changes need to be made to driver function, circle class would not be affected."

(Action, p. 4; *see also* Friedman, p. 513). The Circle class taught by Friedman does not encapsulate an *environment-dependent* communication operation of the algorithm. In fact, the only communication exhibited by the Circle class is that between itself and an algorithm that calls its functions. (Friedman, p. 511). It will be readily understood by anyone of skill in the art that such communication between two entities written in the same programming language is not environment-dependent. Thus, for at least this additional reason, the rejection of claim 2 should not be sustained.

Claim 3:

Claim 3 recites "wherein said at least one environment-dependent communication operation comprises all environment-dependent communication operations of said algorithm." This claim is further patentable over the prior art for at least the same reasons given above with regard to claim 2. Friedman simply does not teach or suggest "environment-dependent" communication operations.

Claim 15:

Claim 15 recites:

> A computer readable medium storing thereon computer executable instruction code, said code when executed by a processor of a computer causes said processor to:
> execute a *first module encapsulating a computer algorithm except at least one communication operation of said algorithm*; and
> execute a *second module encapsulating said at least one communication operation of said algorithm, such that said at least one communication operation is available to said first module*,

wherein *said second module encapsulates at least one environment-dependent communication operation of said algorithm* and is configured to communicate with a communicating partner.

(Emphasis added).

In contrast, Friedman does not teach or suggest the subject matter of claim 15. Specifically, as has been amply demonstrated above, Friedman fails to teach or suggest "a first module encapsulating a computer algorithm except at least one communication operation of said algorithm" and "a second module encapsulating said at least one communication operation of said algorithm, such that said at least one communication operation is available to said first module." (Claim 15).

"A claim is anticipated [under 35 U.S.C. § 102] only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 2 U.S.P.Q.2d 1051, 1053 (Fed. Cir. 1987) (emphasis added). See M.P.E.P. § 2131. For at least these reasons, the rejection based on Friedman of claim 15 and its dependent claims should not be sustained.

(2) Claims 4 and 6-14 are patentable over Friedman:

Claims 4 and 6-14 are each patentable over Friedman for at least the same reasons given above with respect to the patentability of independent claim 1 over Friedman. For at least these reasons and the following additional reasons, the rejection of claims 4 and 6-14 should not be sustained.

Claim 4:

Claim 4 recites "executing a third module encapsulating another communication operation of said algorithm." The final Office Action asserts that this subject matter is obvious over Friedman because "Friedman disclosed the advantages of using subordinate functions to separately implement a program," specifically citing to Friedman's separate implementation of draw_circle, draw_triangle, and draw_intersect functions. (Action, p. 5) (citing to Friedman, pp. 113-119).

Nevertheless, the final Office Action has failed to demonstrate that Friedman disclosed the advantages to separately encapsulating only communication functionality or any other reason that would suggest the subject matter of claim 4 to one having ordinary skill in the art. Thus, the final Office Action has not established that it would be obvious to one having ordinary skill in the art to implement multiple modules encapsulating only communication operations of an algorithm.

The Supreme Court recently addressed the issue of obviousness in *KSR Int'l Co. v. Teleflex Inc.*, 127 S.Ct. 1727 (2007). The Court stated that the *Graham v. John Deere Co. of Kansas City*, 383, U.S. 1 (1966), factors still control an obviousness inquiry. Under the analysis required by *Graham* to support a rejection under § 103, the scope and content of the prior art must first be determined, followed by an assessment of the differences between the prior art and the claim at issue in view of the ordinary skill in the art. In the present case, the scope and content of the prior art, as evidenced by Friedman, did not include the claimed subject matter, particularly multiple modules encapsulating only communication operations of an algorithm.

The differences between the cited prior art and the claimed subject matter are significant because the claimed subject matter provides features and advantages not known or available in the cited prior art. Moreover, the Examiner has not shown that all of the features of claim 4 are taught or suggested by the prior art, and therefore has failed to show the subject matter of claim 4 to be *prima facie* obvious. For at least these additional reasons, the rejection of claim 4 should not be sustained under 35 U.S.C. § 103 and *Graham*.

Claims 13-14:

Claim 13 recites "wherein said at least one method of communication comprises a method of communicating data from said first data object to said communication partner" and claim 14 recites "wherein said at least one method of communication comprises a method of communicating data from said communicating partner to said second data object." Regarding this subject matter, the final Office Action states that "Friedman teaches outputting data from class circle member function to user and passing parameter [sic] (communicating data) from object my_circle to various member functions (communication partner) to communication [sic] among multiple classes." (Action, p. 7) (citing to Friedman, Figs. 11.7-11.8 and p. 706).

The Examiner's logic here contradicts itself. According to the final Office Action, the various member functions read on the communication partner recited in the claims. (*Id.*). However, these cited member functions perform the same functionality as the driver function which the final Office Action also asserts to read on the second module of claim 1. (Action, p. 4; *see also* Friedman, Figs. 11.7-11.8 and p. 706). Therefore, following the Examiner's reasoning, the same block of code can simultaneously function as both a portion of an algorithm and an

external communication partner to the algorithm. Appellant rejects this faulty reasoning in the final Action and maintains that the final Office Action has not demonstrated that Friedman teaches or suggests the subject matter of claims 13-14.

According to the Supreme Court, the Examiner is required to provide an explicit analysis as to how the cited prior art teaches or suggests all the features of a claim. "To facilitate review, this [the Examiner's] analysis should be made explicit." (*KSR International Co. v. Teleflex, Inc.*, 550 U.S. ___ (2007). Therefore, under the standard of *KSR*, no *prima facie* case of obviousness has been made as to claims 13-14. For at least this reason, the rejection of claims 13-14 should not be sustained.

In view of the foregoing, it is submitted that the final rejection of the pending claims is improper and should not be sustained. Therefore, a reversal of the Rejection of January 21, 2009 is respectfully requested.

Respectfully submitted,

DATE: May 18, 2009

/Steven L. Nichols/
Steven L. Nichols
Registration No. 40,326

Steven L. Nichols, Esq.
Managing Partner, Utah Office
Rader Fishman & Grauer PLLC
River Park Corporate Center One
10653 S. River Front Parkway, Suite 150
South Jordan, Utah 84095
(801) 572-8066
(801) 572-7666 (fax)

## VIII. CLAIMS APPENDIX

1. (previously presented)A method of executing a computer algorithm, comprising:

executing a first module encapsulating said computer algorithm except at least one communication operation of said algorithm;

executing a second module encapsulating said at least one communication operation of said algorithm, such that said at least one communication operation is available to said first module; and

instantiating at least one data object for encapsulating data communicated between said first module and a communicating partner, each one of said at least one data object being an instance of a data class, said data communicated between said first module and said communicating partner being accessible by said first module.

2. (original) The method of claim 1, wherein said at least one communication operation comprises at least one environment-dependent communication operation of said algorithm.

3. (original) The method of claim 2, wherein said at least one environment-dependent communication operation comprises all environment-dependent communication operations of said algorithm.

4. (original) The method of claim 1, further comprising executing a third module encapsulating another communication operation of said algorithm.

5. (cancelled)

6. (previously presented)The method of claim 1, wherein data from said first module is encapsulated in a first data object being an instance of a first data class, and data to said first module is encapsulated in a second data object being an instance of a second data class.

7. (original) The method of claim 6, wherein said second module comprises a communication object, said communication object being an instance of a communication class.

8. (original) The method of claim 7, wherein said first module comprises a command object, said command object being an instance of a command class.

9. (original) The method of claim 8, wherein each one of said classes implements one of a plurality of protocols of a framework, such that instances of said classes are compatible with each other.

10. (original) The method of claim 9, wherein said framework is a Java framework and each one of said plurality of protocols is respectively encapsulated in an interface.

11. (original) The method of claim 10, wherein said command class implements a command interface, said command interface defining at least one method of executing, said method of executing taking an indicator of said communication object as a parameter, thereby an operation of said communication object is available to said command object.

12. (original) The method of claim 11, wherein said communication class implements a communication interface, said communication interface defining at least one method of communication.

13. (original) The method of claim 12, wherein said at least one method of communication comprises a method of communicating data from said first data object to said communication partner.

14. (original) The method of claim 13, wherein said at least one method of communication comprises a method of communicating data from said communicating partner to said second data object.

15. (previously presented) A computer readable medium storing thereon computer executable instruction code, said code when executed by a processor of a computer causes said processor to:

execute a first module encapsulating a computer algorithm except at least one communication operation of said algorithm; and

execute a second module encapsulating said at least one communication operation of said algorithm, such that said at least one communication operation is available to said first module,

wherein said second module encapsulates at least one environment-dependent communication operation of said algorithm and is configured to communicate with a communicating partner.

16-25. (cancelled)

26. (previously presented)The computer system of claim 15, wherein each one of said first and second module codes implements a common protocol so that said first and second module codes are compatible.

## IX. Evidence Appendix

None

## X. Related Proceedings Appendix

None